
Init Minix

MINIX

MINIX

Raccoons

they do not wash their food

Raccoons

they do not wash their food

they prefer to live in urban areas

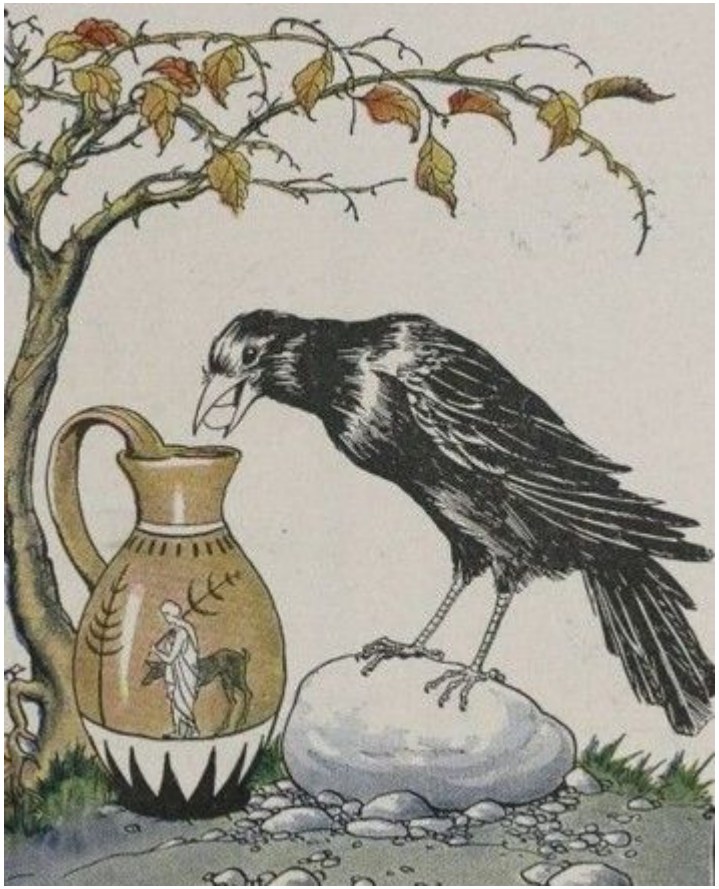
Raccoons

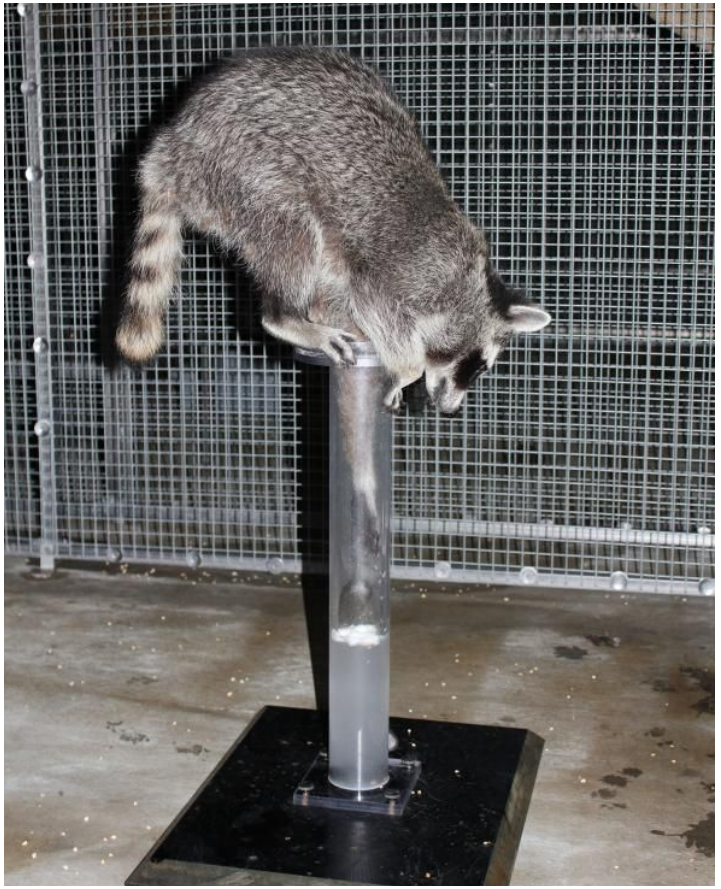
they do not wash their food

they prefer to live in urban areas

they are nocturnal animals

Raccoons And The Aesop's Fable Test





https://news.nationalgeographic.com/2017/10/animals-intelligence-raccoons-birds-aesops/


★ MINIX
★ Configuration
★ Who Are Users?

Why?

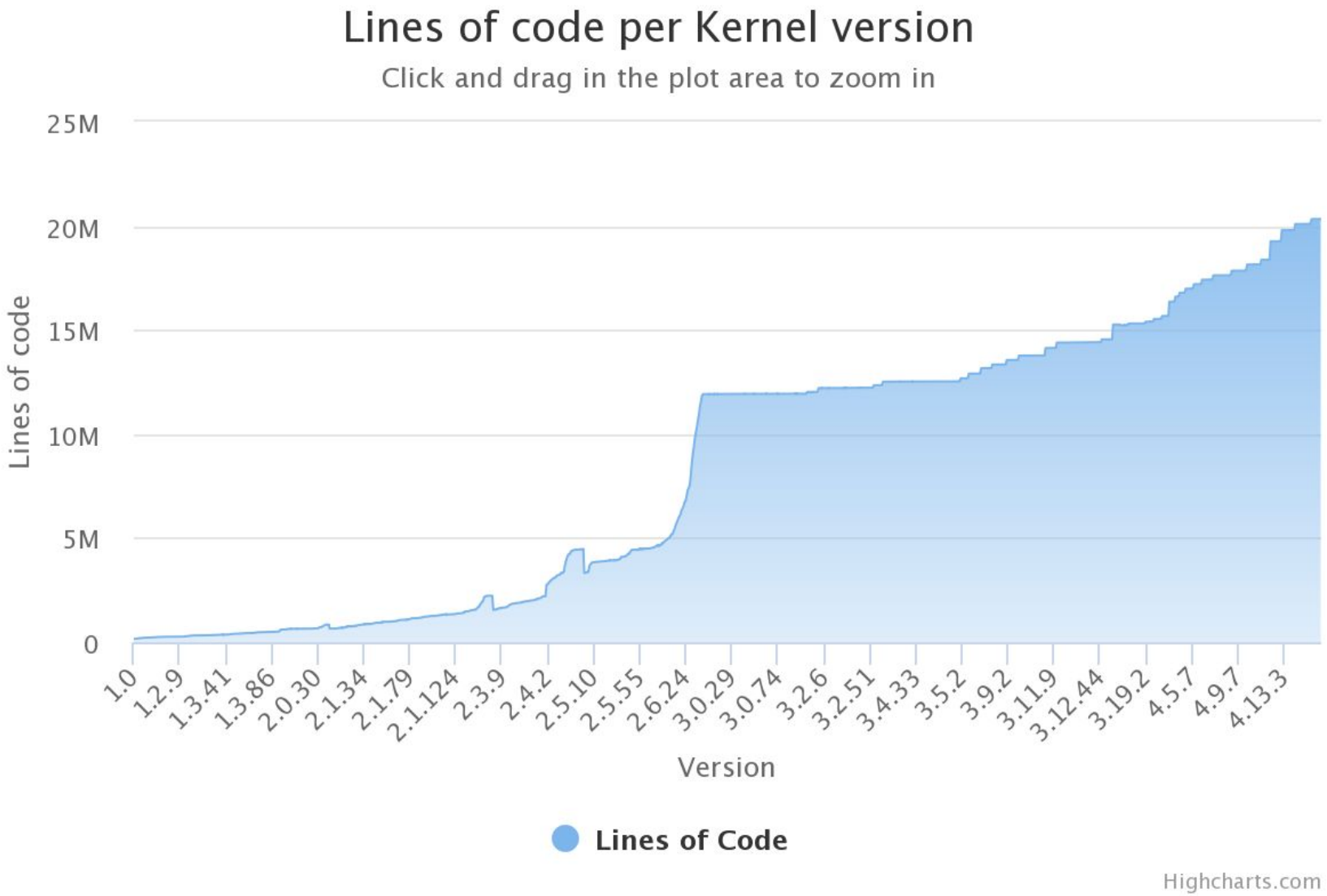
always open-source

compliant with standards like POSIX

modular architecture

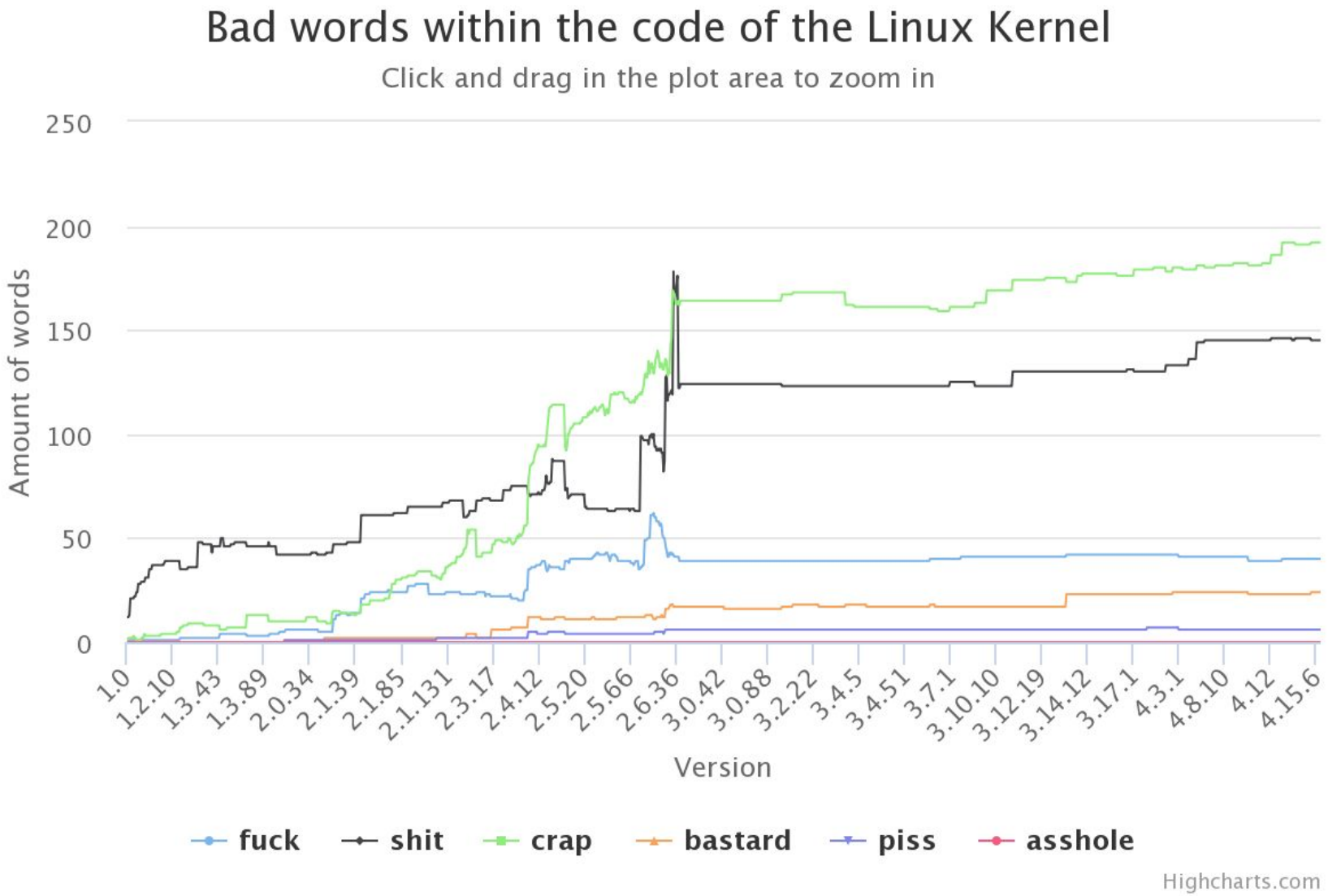
printable amount of code (30k LOC)

Why?



https://www.linuxcounter.net/statistics/kernel

Why?



https://www.linuxcounter.net/statistics/kernel

MINIX docs

MINIX community

MINIX-assignments: workflow

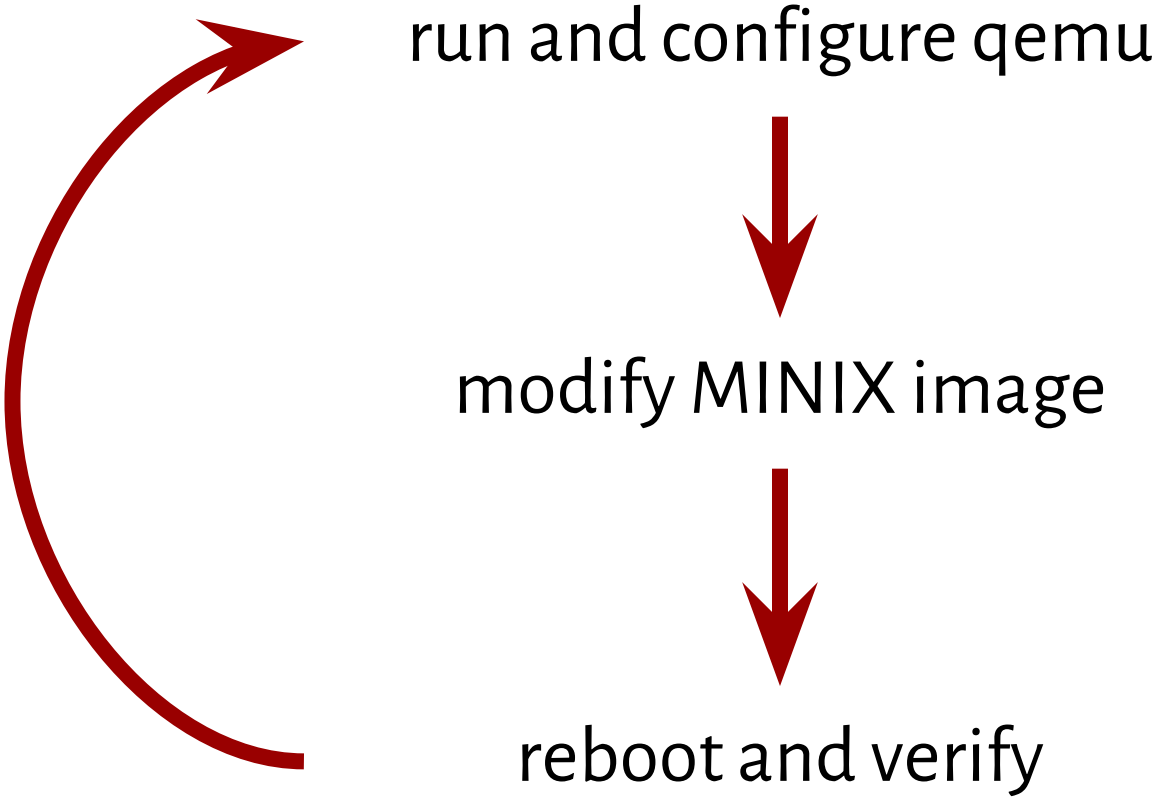
run and configure qemu

MINIX-assignments: workflow

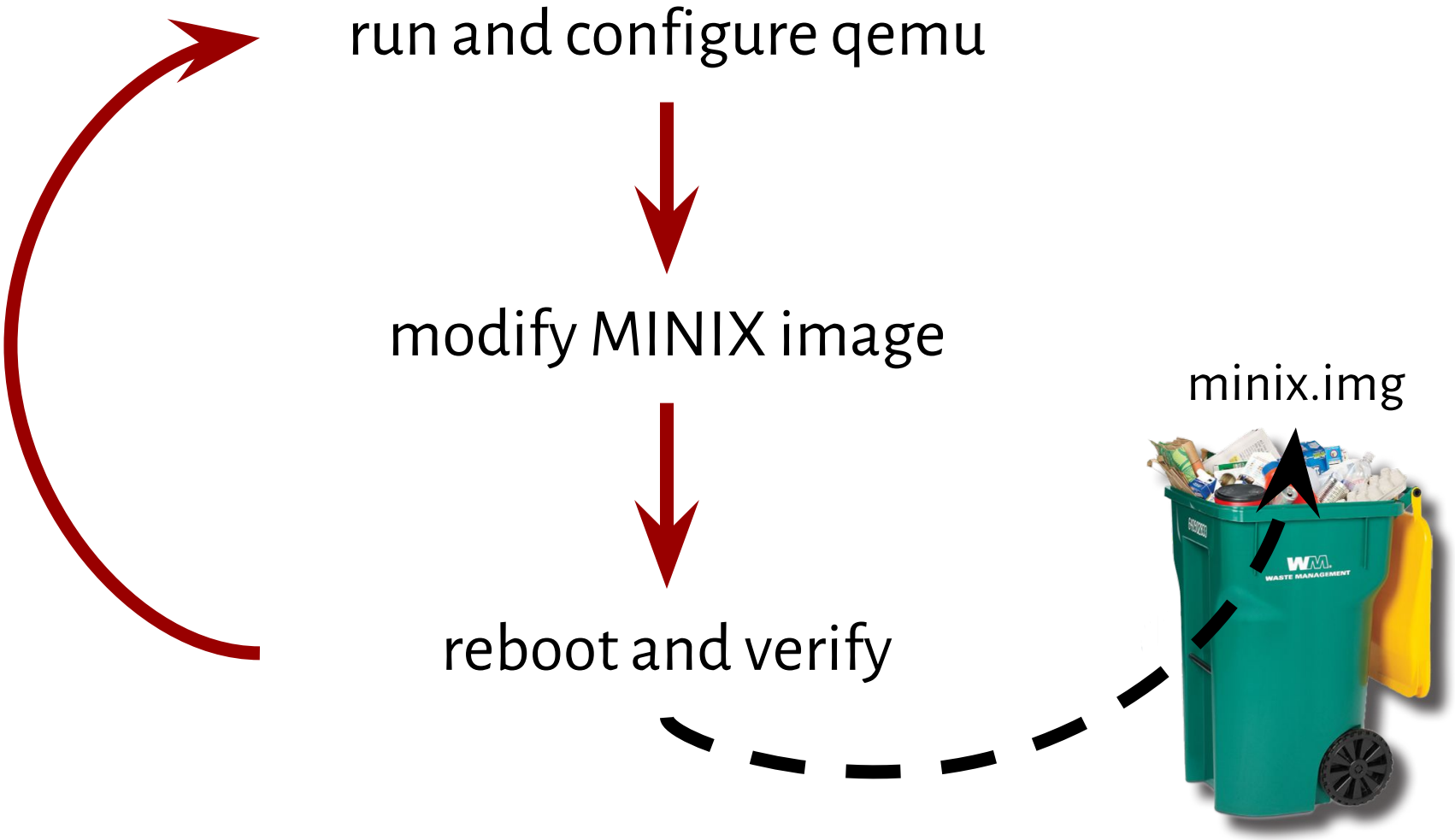
run and configure qemu

modify MINIX image

MINIX-assignments: workflow



MINIX-assignments: workflow



★ MINIX

★ **Configuration**

★ Who Are Users?

How to share and protect the code?

★ SSHFS (SSH File System)

```
$ sudo apt-get install sshfs
```

```
$ sshfs root@localhost:<sciezka_do_katalogu_na_minixie>
<sciezka_do_punktu_montowania_na_hoscie> -p 10022
```

How to share and protect the code?

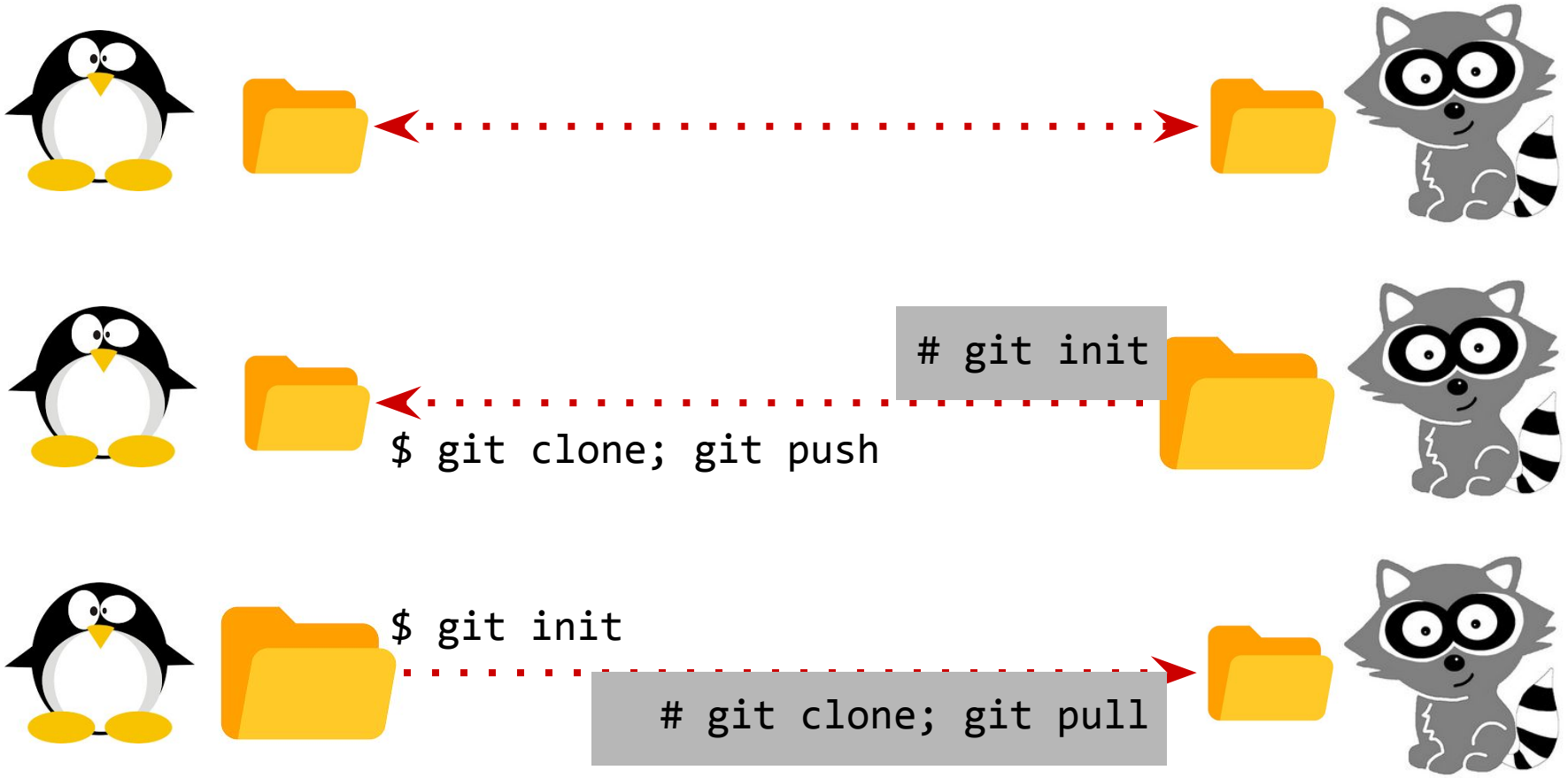
★ SSHFS (SSH File System)


```
$ sudo apt-get install sshfs
```

```
$ sshfs root@localhost:<sciezka_do_katalogu_na_minixie>
<sciezka_do_punktu_montowania_na_hoscie> -p 10022
```

How to share and protect the code?

★ Git repository



What I expect - scripts

- ★ **create image and run QEMU**
- ★ exchange RSA keys
- ★ apply modifications

Create the image and run QEMU

```
#!/bin/bash

qemu-img create -f qcow2 -o backing_file=backup_minix.img
minix.img

qemu-system-x86_64 -curses -drive file=minix.img -enable-kvm
-localtime -net user,hostfwd=tcp::10022-:22 -net
nic,model=virtio -m 1024M
```

QEMU - binary translation for various CPUs

binary opcodes for a specific CPU

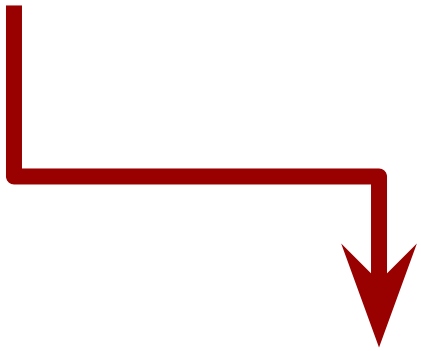
software functions
compatible with the host's OS

translation to valid opcodes

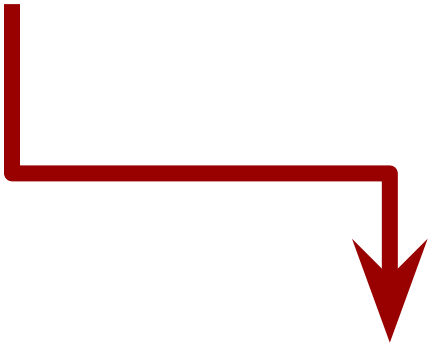
so slow and complicated...

KVM - the idea in a nutshell

span a thread



recognize native instructions



execute them directly

KVM - the idea in a nutshell

When working together, KVM arbitrates access to the CPU and memory, and QEMU emulates the hardware resources (hard disk, video, USB, etc.).

When working alone, QEMU emulates both CPU and hardware.

https://serverfault.com/a/208694

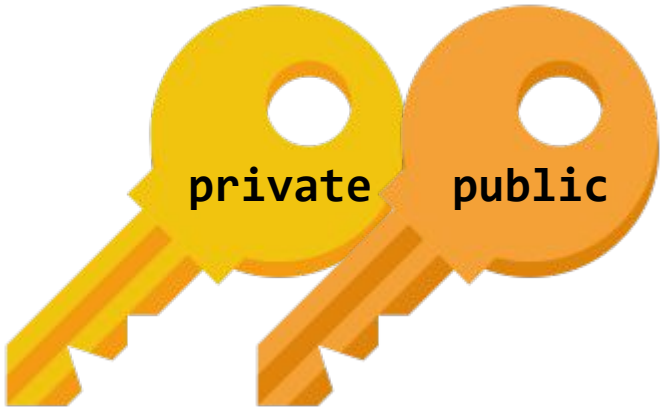
What I expect - scripts

- ★ create image and run QEMU
- ★ **exchange RSA keys**
- ★ apply modifications

Exchange RSA keys

https://help.ubuntu.com/community/SSH/OpenSSH/Keys

```
mkdir ~/.ssh
chmod 700 ~/.ssh
ssh-keygen -t rsa
```



Exchange RSA keys

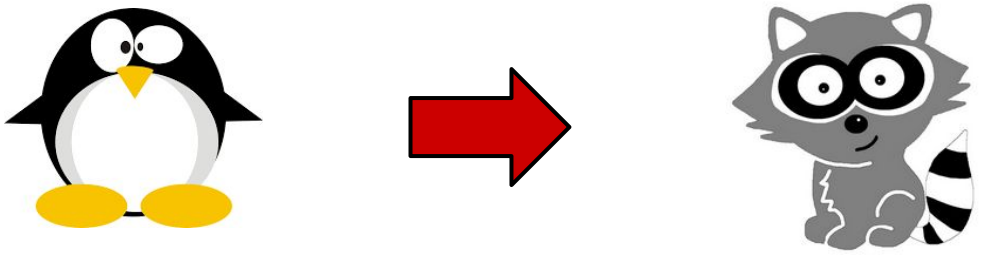
https://help.ubuntu.com/community/SSH/OpenSSH/Keys

```
mkdir ~/.ssh
chmod 700 ~/.ssh
ssh-keygen -t rsa

ssh-copy-id root@localhost -p 10022
```

Use SSH

https://help.ubuntu.com/community/SSH/OpenSSH/Keys

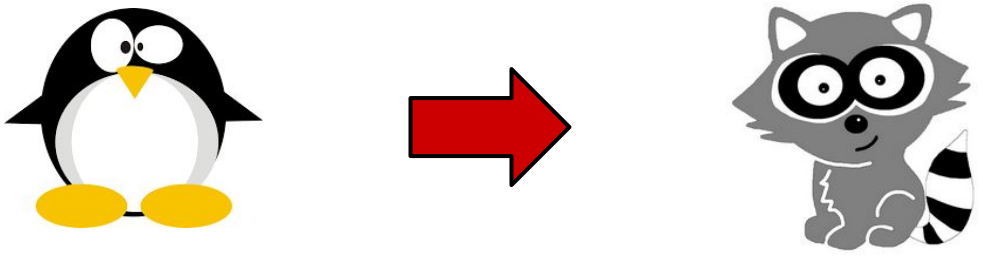


```
$ ssh root@localhost -p 10022

$ scp -P 10022 <path_to_file> root@localhost:<path_to_file>
```

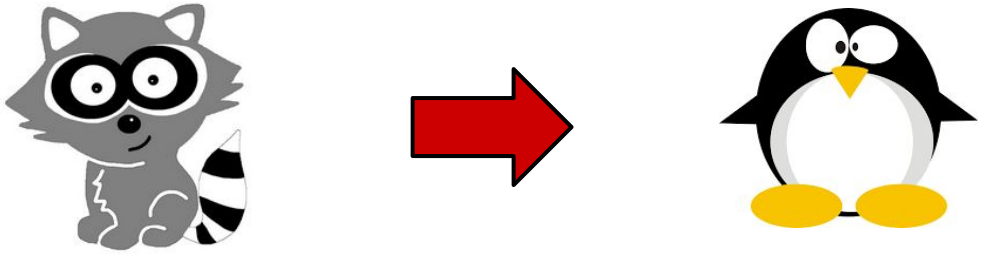
Use SSH

https://help.ubuntu.com/community/SSH/OpenSSH/Keys



```
$ ssh root@localhost -p 10022

$ scp -P 10022 <path_to_file> root@localhost:<path_to_file>
```



```
# ssh <host_username>@10.0.2.2
```

Use git

```
# git clone ssh://<host_username>@10.0.2.2<repo_na_linuxie>
```

Use git

```
# git clone ssh://<host_username>@10.0.2.2<repo_na_linuxie>
```

```
$ ssh -p 10022 root@localhost 'ls /root/'
```

What I expect - scripts

- ★ create image and run QEMU
- ★ exchange RSA keys
- ★ apply modifications

What I **expect** - scripts

- ★ create image and run QEMU
- ★ exchange RSA keys
- ★ apply modifications

What I **expect** - scripts

```
#!/usr/bin/expect -f

spawn ssh HOSTNAME

expect "Login:"

send "jakis_login12\r"

expect "Password:"

send "jakies_haslo\r"

interact
```

https://linux.die.net/man/1/expect


★ MINIX
★ Configuration
★ Who Are Users?

Users

configuration

/etc/profile

~/.profile

http://bencane.com/2013/09/16/understanding-a-little-more-about-etcprofile-and-etcbashrc/

https://shreevatsa.wordpress.com/2008/03/30/zshbash-startup-files-loading-order-bashrc-zshrc-etc/

Users

configuration

permissions

/etc/profile

~/.profile

http://bencane.com/2013/09/16/understanding-a-little-more-about-etcprofile-and-etcbashrc/

https://shreevatsa.wordpress.com/2008/03/30/zshbash-startup-files-loading-order-bashrc-zshrc-etc/

Users

configuration

permissions

/etc/profile

~/.profile

http://bencane.com/2013/09/16/understanding-a-little-more-about-etcprofile-and-etcbashrc/

https://shreevatsa.wordpress.com/2008/03/30/zshbash-startup-files-loading-order-bashrc-zshrc-etc/

Users

configuration

permissions

/etc/profile

~/.profile

chmod

chown, chgrp

http://bencane.com/2013/09/16/understanding-a-little-more-about-etcprofile-and-etcbashrc/

https://shreevatsa.wordpress.com/2008/03/30/zshbash-startup-files-loading-order-bashrc-zshrc-etc/

Zadanie D2

Na swoim obrazie systemu MINIX utwórz nową grupę **‘friends’**.

Dodaj dwóch nowych użytkowników o loginach **‘alice’** i **‘bob’**.

Dla każdego z nich utwórz katalog domowy i ustaw grupę **‘friends’** jako główną. Obejrzyj pliki **/etc/passwd** i **/etc/group**, żeby zweryfikować poprawność rozwiązania. Następnie zaloguj się na konto alice, używając polecenia **login**.

Zadanie D3

Korzystając z przygotowanej konfiguracji, zmodyfikuj pliki **~/.profile roota** i **alice** w taki sposób, aby:

- tuż po zalogowaniu **root** automatycznie tworzył katalog **/tmp/secret** (jeśli nie istnieje), należący do grupy friends;
- nikt poza grupą friends i właścicielem nie mógł ani zobaczyć, ani zmodyfikować zawartości tego katalogu;
- wszystkie pliki utworzone gdziekolwiek przez **alice** mogły być modyfikowane przez jej grupę główną (wskazówka: umask)

Zadanie D2

```
# group add friends

# useradd -m -g friends alice    # utwórz użytkownika alice
# passwd alice                   # ustal hasło dla użytkownika alice

# useradd -m -g friends bob      # utwórz użytkownika bob
# passwd bob                     # ustal hasło dla użytkownika bob
```

Zadanie D3

Korzystając z przygotowanej konfiguracji, zmodyfikuj pliki **~/.profile roota** i **alice** w taki sposób, aby:

- tuż po zalogowaniu **root** automatycznie tworzył katalog **/tmp/secret** (jeśli nie istnieje), należący do grupy friends;
- nikt poza grupą friends i właścicielem nie mógł ani zobaczyć, ani zmodyfikować zawartości tego katalogu;
- wszystkie pliki utworzone gdziekolwiek przez **alice** mogły być modyfikowane przez jej grupę główną (wskazówka: umask)

Zadanie D2

```
# group add friends

# useradd -m -g friends alice    # utwórz użytkownika alice
# passwd alice                   # ustal hasło dla użytkownika alice

# useradd -m -g friends bob      # utwórz użytkownika bob
# passwd bob                     # ustal hasło dla użytkownika bob
```

Zadanie D3

```
/root/.profile
--------------
# dodajemy następujące polecenia
if [ ! -d "/tmp/secret" ]; then
    mkdir /tmp/secret
    chgrp friends /tmp/secret
    chmod 770 /tmp/secret
fi
```

```
/home/alice/.profile
--------------------
# modyfikujemy domyślne ustawienie
umask
umask 012
```

For more flexibility see: https://wiki.archlinux.org/index.php/Access_Control_Lists

Assignment #2

Expected in:

https://svn.mimuw.edu.pl/repos/SO/studenci/<login>/zadanie2

with a proper filename and no additional files

Expected by:

26 March 2018, 8 p.m.

Optionally, show it to me by:

23 March 2018, 8 p.m.